LEGO® TC logo

LEGO Dacta
Educational Products
555 Taylor Road
Enfield, CT 06082

LEGO Australia Pty. Ltd.
P.O. Box 639
Lane Cove
N.S.W. 2066

LEGO dacta

ISBN 0-914831-69-0
198324
Printed in U.S.A.

LEGO® TC logo

Making Machines

LEGO dacta

No part of the documents contained herein may be reproduced, stored in retrieval systems or transmitted, in any form or by any means, photocopying, electronic, mechanical, recording or otherwise, without the prior approval in writing from the LEGO Group.

2nd revision, January 1989

ISBN 0-914831-69-0

LEGO® and the LEGO logo® are registered trademarks of INTERLEGO AG. © 1987, 1989 LEGO Group.

Apple and the Apple logo are registered trademarks of Apple Computer, Inc.

IBM is a registered trademark of International Business Machines Corporation.

MS-DOS is a registered trademark of Microsoft Corporation.

Making Machines

Introduction

Machines are everywhere. We use machines at home, at work and at play.

Indeed, our ability to make machines is one thing that makes us different from other animals.

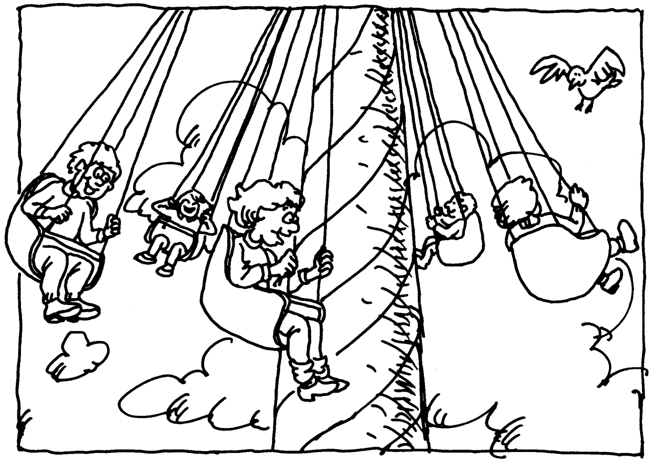
Machines can help us in many ways. Some machines do things that we would find boring or dangerous if we had to do them ourselves.

Other machines **amplify** our abilities. When we use a car, we become faster. When we use a crane, we become stronger.

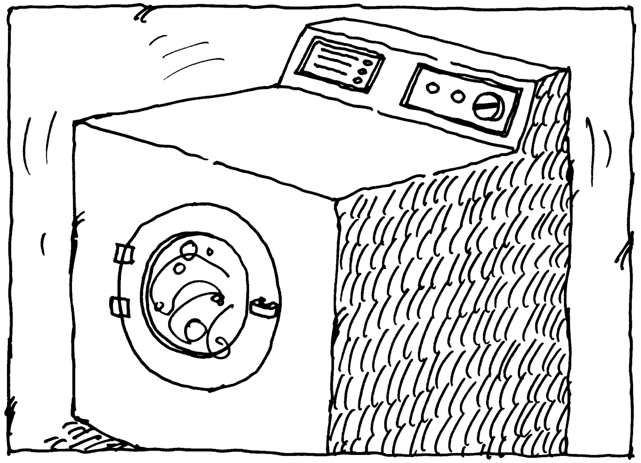
You can use a computer to make machines even more useful. Using a computer, you can **program** a machine to do a complex series of tasks. Then, you can easily change the program to make the machine do something different.

This booklet has three projects: A merry-go-round, a washing machine, and a conveyer belt. Each project suggests ways to program the machines you build.

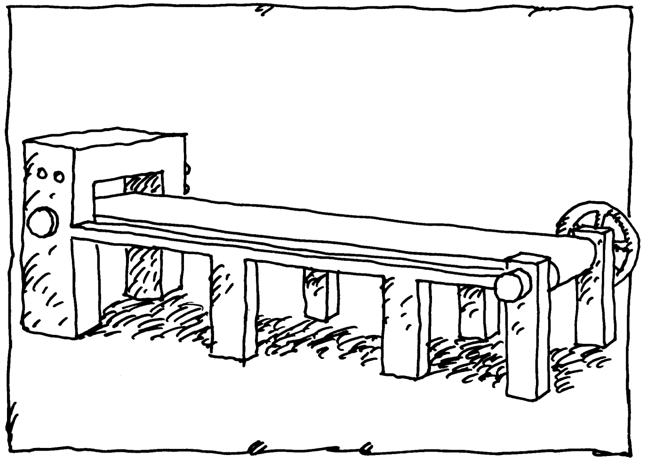
These three projects should help you get started. Of course, there are many other machines that you can build and program with LEGO® TC logo. So use your imagination. Think of a machine you have seen, then try to build it. Or invent a new machine of your own!



Play



Home



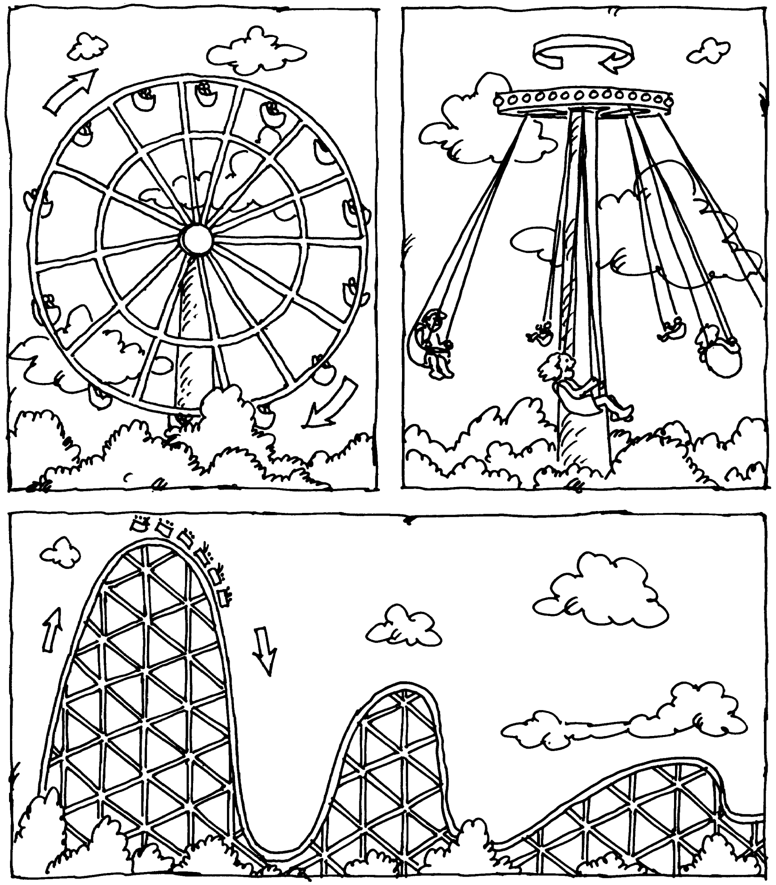
Work

Project 1

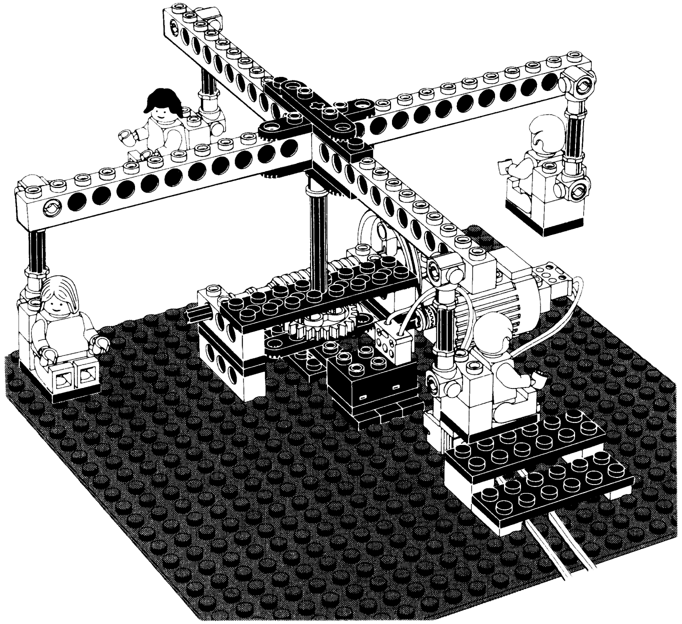
Machines at Play: Merry-Go-Round

Amusement parks are full of machines: Ferris wheels, roller coasters, merry-go-rounds. These machines are fun to ride because of the way they move. Some move you up and down. Others spin you around.

Activity Card 6 shows how to build a LEGO® merry-go-round.



Starting the Ride



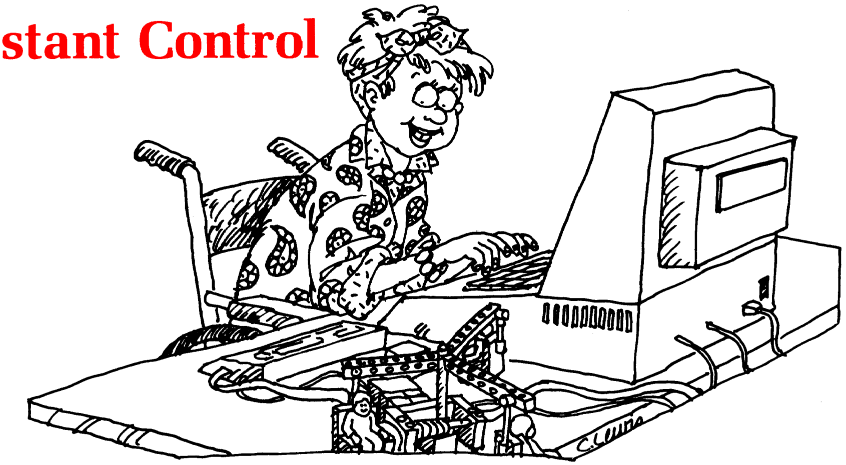
Plug the motor into port A, and type `talkto "A` Now try `onfor 100`. How many times does the merry-go-round go around? How can you make the merry-go-round go around exactly once?

You can change the speed of the merry-go-round using the LEGO® TC logo command `setpower`. Try these commands:

`on`	◀ Turn on the motor
`setpower 3`	◀ Set the motor power to three
`setpower 6`	◀ Set the motor power to six

To return to "normal" speed, type `setpower 7`.

Instant Control



You can write an "instant" procedure to control the merry-go-round. This type of procedure lets you control the machine by pressing a single key on the keyboard.

Let's say that you want the merry-go-round to **S**tart every time you press the S key. And you want the machine to turn **O**ff every time you press the O key, and you want it to **R**everse direction every time you press the R key.

You can write the procedure like this:

```
to instant
talkto "A
name readchar "key
if :key = "S [on]
if :key = "O [off]
if :key = "R [rd]
instant
end
```

◀ Don't forget the quotes(")

◀ Don't forget the colon (:)

The LEGO® TC logo procedure `readchar` stands for "read character." It waits until you type a character (a letter or a number or some other symbol) on the keyboard. Then it reports which character you typed.

The line `name readchar "key` gives the name `key` to whatever character you type. If you type the letter S, then S will have the name `key`. (You need the quotes to tell the computer that you are talking about the **word** `key`, not the **procedure** `key`.)

When you see a colon, substitute the words "the thing that has the name." So `:key` means "the thing that has the name `key`."

The line `if :key = "S [on]` checks to see if S is the thing that has the name `key`. If so, the computer turns on the motor.

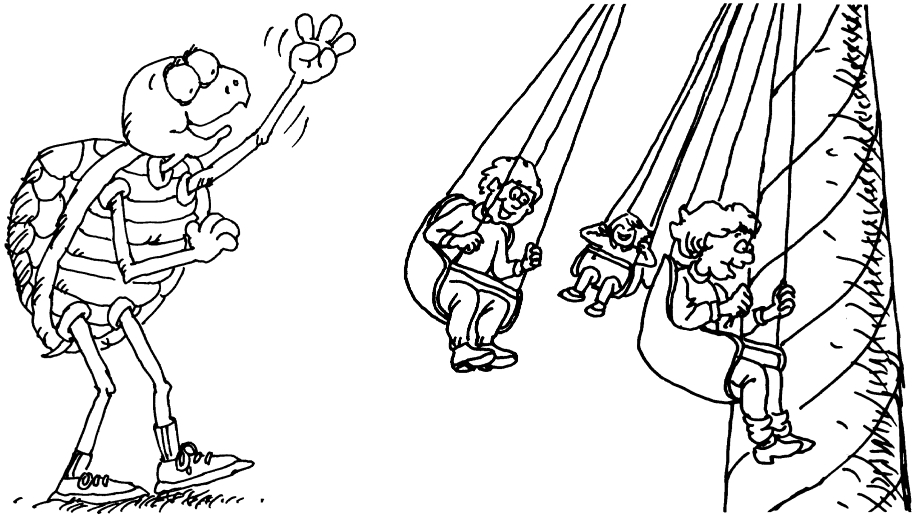
Similarly, `if :key = "O [off]` checks to see if O is the thing that has the name `key`. If so, the computer turns off the motor.

Challenges

1. Write a new "instant" procedure that makes the merry-go-round turn once if you type the number 1, twice if you type the number 2, and so on.

2. Write another instant procedure that lets you control the speed of the merry-go-round. If you type 4, the procedure should set the "power" to 4.

Counting the Turns



You have already seen how to use `onfor` to control the merry-go-round.

There is a problem with this approach. Let's say that the merry-go-round revolves exactly once when you type `onfor 10`. What will happen the next time you type `onfor 10`? Will the merry-go-round revolve exactly once?

Maybe, but maybe not. Imagine that a LEGO® piece has slipped a bit, increasing the friction in the system. When you type `onfor 10` again, the merry-go-round will not go as far as before.

Another problem: Some motors work "better" in one direction than the other. So `onfor 10` in one direction is not the same as `onfor 10` in the other direction.

Using a touch sensor, you can control the merry-go-round with greater precision (See Activity Card 6). The touch sensor is pressed four times each time the merry-go-round makes one turn. (Make sure that the four "arms" are adjusted to hit the touch sensor.)

The procedure `oneturn` should make the merry-go-round turn exactly once:

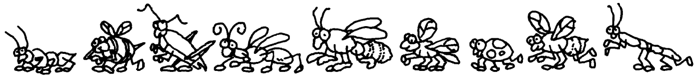
```
to oneturn
listento 6
resetc
talkto "A
on
waituntil [counter = 4]
off
end
```

Plug the sensor into port 6. Then try `oneturn`.

There are stairs leading up to the merry-go-round. At the start of the ride, you might want each chair to stop at the top of the stairs to let a person get on.

```
to geton
listento 6
talkto "A
repeat 4 [on
          waituntil [sensor?]
          wait 6
          off
          wait 40]
end
```

BUG BOX



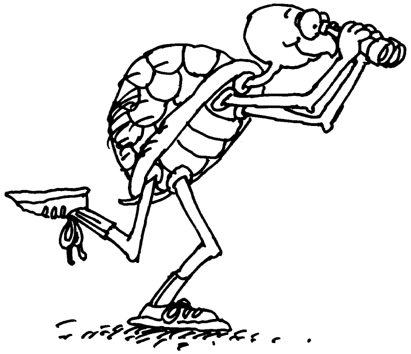
The wrong side of the wheel is facing the sensor.
You left a space in `oneturn` or `geton`.

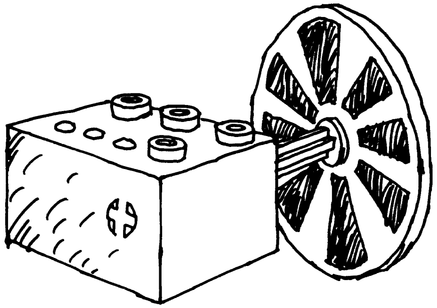
Challenges

1. Write a procedure `ride` that controls the entire ride. Here's one approach: First, the merry-go-round stops at each chair to let people get on. Then it turns a given number of times. Finally it stops at each chair to let people get off.

2. Write a procedure `turns` that takes a number as an input. If you type `turns 4`, the merry-go-round should go around four times. (Hint: See page 52 of *Getting Started*.)

3. Replace the touch sensor with an optosensor and counting wheel, as shown on the back page of Activity Card 6. Change the `oneturn` procedure so that it works with the optosensor.





Merry-Go-Round Music

You can use the LEGO® TC logo primitive `tone` to make the computer play music while the merry-go-round turns.

Here's an example:

```
to merry
talkto "A
on
tone 330   9
tone 311   3
tone 330   3
tone 392   3
tone 523  12
tone 494   3
tone 523   3
tone 587   3
tone 523   3
tone 494   3
tone 523   3
tone 330   3
tone 392   3
tone 494  12
wait 9
merry
end
```

The first input to `tone` sets the **pitch** of the tone. The second input sets the **duration** of the tone.

Try changing the `tone` commands to make a new song for the `merry` procedure.

Project 2

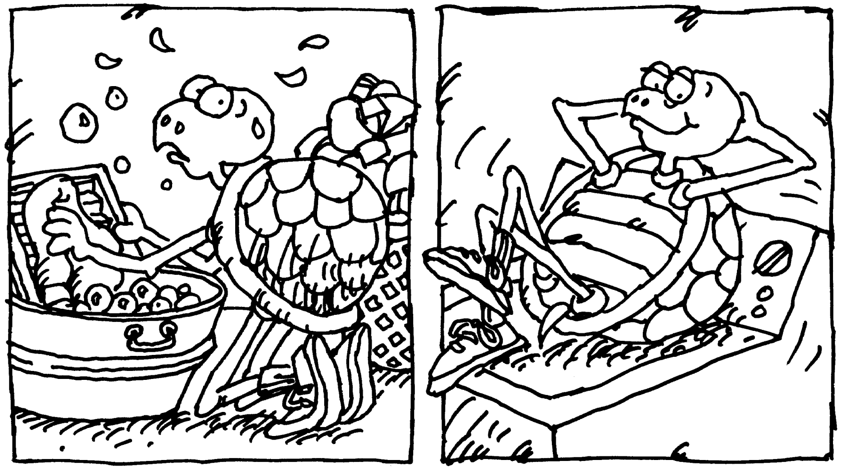
Machines at Home: Washing Machine

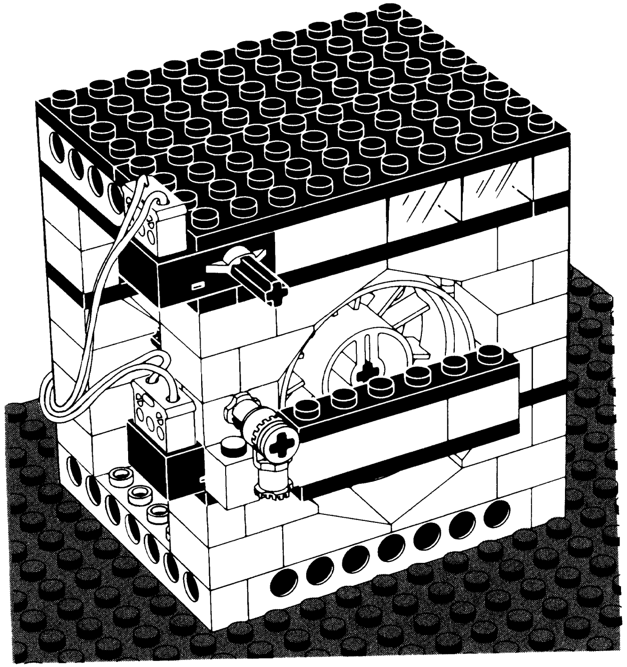
There are many types of machines used in homes: Washing machines, stereos, sewing machines, toasters, dishwashers, televisions, vacuum cleaners, ovens.

These machines make life easier and more enjoyable. Consider the job of washing clothes. Without a washing machine, washing clothes is hard work. You could wash your clothes in a nearby river or stream. Or you could wash your clothes with a washboard in a washbasin (see drawing).

A washing machine makes the job much easier. You still must load the clothes into the machine. But after that, the machine does most of the work. It washes, rinses, and spins them.

Activity Card 7 shows how to build a LEGO® washing machine.





Cycles

In the LEGO® washing machine, the motor turns a large wheel. This wheel acts like the **agitator** in a real washing machine.

The agitator stirs the water and clothes inside the machine, helping clean the clothes. When people use washboards, they must use their own muscles to "agitate" the clothes. In a washing machine, the motor and agitator do this job.

The agitator can turn in several different ways, depending on which **cycle** the machine is running.

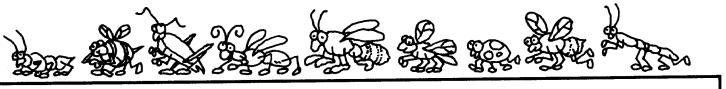
In the **wash** cycle, the agitator turns back and forth. In the **spin-dry** cycle, the agitator turns in just one direction.

Here's a procedure for the wash cycle:

```
to wash
talkto "B
repeat 15 [onfor 5 rd]
end
```

Make sure the motor is plugged into port B. Then try running `wash`.

BUG BOX



The motor is not plugged in.
You are "talking" to the wrong port.
The stop button is pushed in.

Challenges

1. Change the `wash` procedure so that the agitator turns back and forth more quickly.

2. Write a procedure called `spin` for the spin-dry cycle of the washing machine. The agitator should simply spin in one direction.

3. Write a procedure called `laundry` that runs a wash cycle, then a spin-dry cycle. (Hint: Use subprocedures.)

Indicator Lights

Lights can be very useful for displaying information. On a washing machine, lights can show which cycle is running.

The LEGO® washing machine has two lights. You can change the `wash` procedure so that one of the lights is on during the wash cycle. Plug a light into port 0 and a motor into port B.

```
to wash
talkto 0
on
talkto "B
repeat 15 [onfor 5 rd]
talkto 0
off
end
```

Now the light plugged into port 0 will turn on whenever you run `wash`.

Challenge

Change the `spin` procedure so that the other light is on during the spin-dry cycle. Do you have to change anything in the `laundry` procedure?

Starter Switch

The washing machine has two touch sensors. The one next to the indicator lights acts as a **starter switch**. The one next to the door acts as a **safety sensor**.

To help remember which sensor is in which port, you can give names to the ports. Plug the starter switch in port 6 and the safety sensor in port 7. Then type in the Command Center:

```
name 6 "starter
name 7 "safety
```

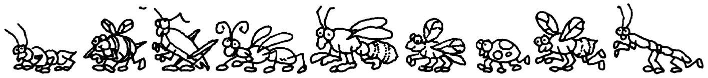
Now `listento :starter` is the same as `listento 6`. Giving names to ports is particularly useful when you use many ports at the same time.

To use the touch sensor as a starter switch, type this procedure:

```
to start
listento :starter
waituntil [sensor?]
laundry
end
```

Run the `start` procedure. The computer waits until you press the touch sensor, then it runs the procedure `laundry`.

BUG BOX



You forgot the quotes(") in the `name` command. You wrote `listento "starter` instead of `:starter`

Advanced Project

The `double.click` procedure uses the starter switch in a trickier way. If you press the touch sensor just once, the machine will run a wash cycle. But if you press the sensor twice in a row (less than two seconds apart), the machine will run a spin-dry cycle.

```
to double.click
listento :starter
waituntil [sensor?]                ◀ Wait until sensor
                                     pressed

waituntil [not sensor?]            ◀ Wait until sensor
                                     released

resett                             ◀ Reset the timer
waituntil [or timer > 20           ◀ Wait 2 seconds or
                     sensor?]        until sensor is
                                     pressed

ifelse timer > 20                  ◀ If no second click,
  [wash][spin]                       run wash

                                     Otherwise, run
                                     spin

end
```

The washing machine doesn't start as soon as you press the sensor. It waits until you release the sensor, then it waits to see if you press the sensor again (within two seconds).

Challenge

Write a procedure that works like this: If you hold down the sensor for less than three seconds, the machine runs `wash`. If you hold down the sensor for more than three seconds, the machine runs `spin`.

Safety Sensor

Safety is an important part of machine design. If a machine is not safe, it is useless, no matter how powerful or efficient the machine might be.

On some washing machines, the agitator stops when you open the door, so that the agitator can't hurt you. You can program the LEGO® washing machine to act the same way. The touch sensor next to the door of the washing machine can act like a safety switch.

The procedure `safe` turns on the machine only when the door is closed.

```
to safe
listento :safety
talkto "B
ifelse sensor? [on][off]
safe
end
```

Make sure the motor and sensor are plugged into the right ports. (If you didn't name the ports before, type `name 7 "safety`.) Then run `safe`.

To stop `safe`, press -S Ctrl-S and type `off`.

To make the `safe` procedure more "readable," you can define a new procedure called `door`. This procedure reports the word **open** or **shut** just as `sensor?` reports **true** or **false**.

```
to door
listento :safety
ifelse sensor? [output "shut]
               [output "open]
end
```

`door` is intended to be used as a subprocedure. It simply reports the word **open** if the door is open or **shut** if the door is shut. (It doesn't print the word on the screen.)

You can use `door` to rewrite `safe` like this:

```
to safe
talkto "B
if door = "open [off]
if door = "shut [on]
safe
end
```

◀ Remember the quotes

Challenge

Write a procedure that uses information from both sensors. One possibility: the washing machine should turn on when you press the touch sensor, then turn off when you open the door.

Advanced Project

What if you wanted the safety sensor to work during the `wash` and `spin` procedures? That is a bit tricky. The procedures must keep checking the sensor while they run, to see whether the door has been opened. You can rewrite `wash` using the subprocedure `turn`:

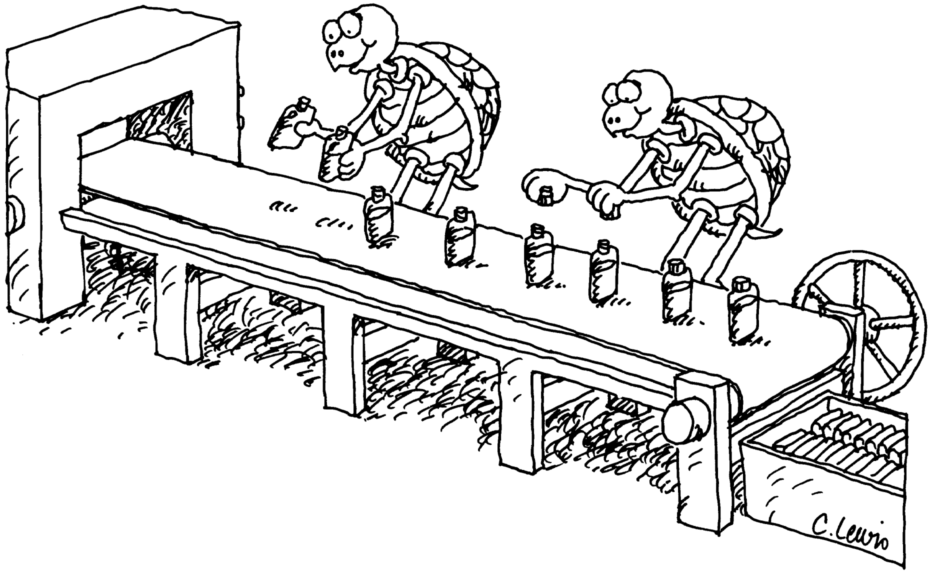
```
to wash
repeat 15 [turn]
end

to turn
listento :safety
waituntil [sensor?]                    ◀ Wait until door is
                                         closed

resett                                 ◀ Reset timer
tto [B 0]                              ◀ Talk to motor in
                                         port B and light
                                         in port 0

on                                     ◀ Turn on agitator
waituntil [or timer > 10               ◀ Wait one second
           not sensor?]                  or until door
                                         is opened
                                         (whichever comes
                                         first)

off                                    ◀ Turn off agitator
rd                                     ◀ Reverse direction
end
```

The `turn` procedure makes sure that the door is closed, then it turns on the agitator for one second — unless someone opens the door before the time is up. The `wash` procedure runs the `turn` procedure 15 times.

You can build and program many other household machines. Here are some ideas:

Project 3

Machines at Work: Conveyor Belt



Companies use machines for two major reasons:

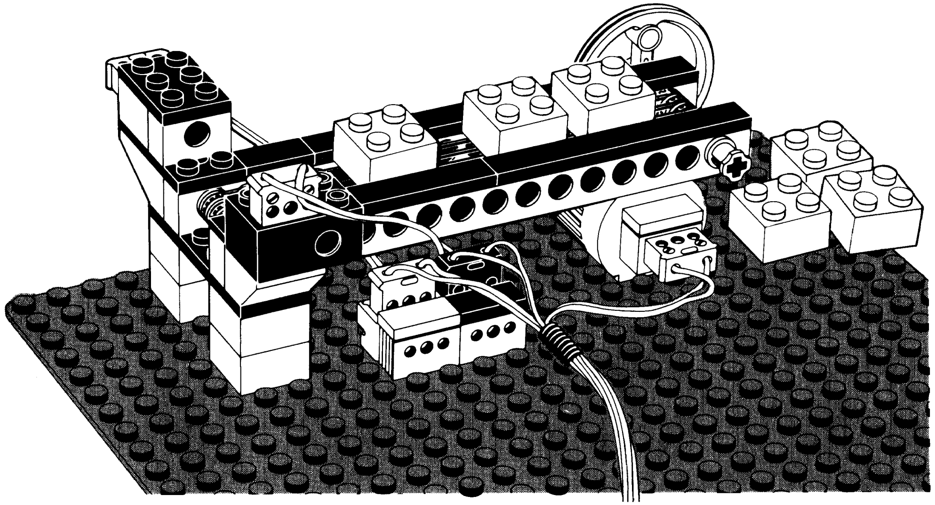
*By using machines, companies can make some products more **efficiently**. That is, they can make the same number of products for less money — or they can make more products for the same amount of money.

*By using machines, companies can make **new products** that would not have been possible without machines.

The conveyor belt is an important machine in many factories. Conveyor belts carry things from one place to another. When people work along a conveyor belt, it is called an **assembly line**.

Assembly lines have changed the way many people work. The conveyor belt carries products from one person to the next, and each person does a small part of the overall job. One person might add some nuts and bolts, while another person paints part of the product.

Activity Card 5 shows how to build a LEGO® conveyor belt.



Counting Objects

You can use an optosensor to count the number of objects that move along the conveyor belt.

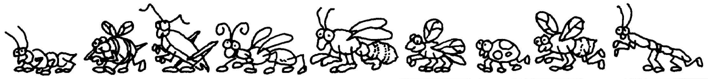
Imagine that the conveyor belt is carrying LEGO® bricks. When a brick passes in front of the sensor, it blocks the light. That increases the value of the sensor's counter. (The counter increases by one every time the sensor changes from `false` to `true`.)

So the value of the counter will tell you how many bricks have moved past the optosensor. `dozen` will wait until 12 objects have moved past the optosensor, then it will turn off the belt.

```
to dozen
talkto "A    ◀ Talk to the motor
on
listento 6
resetc
waituntil [counter = 12]
off
end
```

Turn on the light, then run `dozen`. It is best to use **blue** LEGO® bricks, with the "studs" facing the optosensor.

BUG BOX

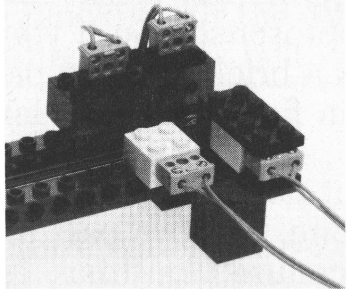


The bricks aren't blocking all the light.
You aren't "listening to" the correct port.

Measuring Length

You can also use optosensors to tell the difference between long objects and short ones.

Let's say you want to detect all objects longer than 2.5 centimeters. Place two optosensors next to one another:



The "eyes" of the optosensors are then about 2.5 centimeters apart. So any object longer than 2.5 centimeters will block **both** optosensors at once.

`sort` will turn off the conveyor belt whenever the sensors "see" an object longer than 2.5 centimeters.

```
to sort
talkto "A                ◀ Talk to the motor
on
listento [6 7]           ◀ Listen to both sensors
waituntil [sensor?       ◀ Wait until both
     = [true true]]        sensors blocked
off
end
```

Before using `sort`, turn on both lights: `talkto [2 3] on`. Be sure to use dark-colored objects (like blue LEGO® bricks).

Advanced Project

You can also use optosensors to calculate the actual **length** of any LEGO® brick that moves along the conveyor belt. In fact, you need only **one** optosensor!

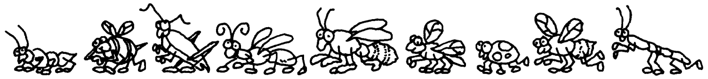
Here is one strategy: Longer bricks take a longer time to move past the optosensor. So if you measure how much time it takes a brick to move past the optosensor, you can figure out how long the brick is.

Start by finding out how much time it takes for a brick, four studs long, to move past the optosensor. You can use a procedure like this:

```
to measure
talkto "A                          ◀ Talk to the motor
on
listento 6
waituntil [sensor?]                ◀ Wait until brick
                                     blocks light
resett                             ◀ Reset the timer
waituntil [not sensor?]            ◀ Wait until light
                                     not blocked
show timer                         ◀ Print the time
off
end
```

Turn on the LEGO® light and run `measure`. Put a four-stud brick on the belt. Let's say that the computer prints `2`. So it took two ticks of the timer (that is two tenths of one second) for the brick to move past the sensor. A two-stud brick would take one tick to move past the sensor. Similarly, an eight-stud brick should take four ticks to move past the sensor. Try it.

BUG BOX



The brick is too light-colored or shiny.
The brick isn't blocking all the light.
The brick is "tilted" on the belt.

Challenges

1. Write a procedure called `sorter`. The procedure should stop the belt whenever a brick longer than a certain length moves down the conveyor belt. `sorter` should take a "cutoff length" (in number of studs) as an input. For example, if you type `sorter 2`, the conveyor belt should move until you put a brick longer than two studs on the belt.

2. Add a second conveyor belt. (See the back page model on Activity Card 5.) Write a procedure that sorts bricks according to length. If a brick is three studs or longer, the second conveyor belt should move the brick in one direction. If the brick is shorter than three studs, the second belt should move it in the other direction.

You should end up with two piles, one with bricks three studs or longer, one with bricks two studs or shorter.

Glossary List

The page numbers in the definitions refer to the ***Getting Started*** *book pages where the words are first described.*

axle: A thin rod that connects to the middle of a wheel (or gear). The wheel and axle rotate together.

Command Center: The place where you type commands for the computer to carry out. The Command Center is at the bottom of the computer screen (page 11).

cursor: A small blinking light that indicates the current position on the computer screen (page 11).

debugging: The act of finding and fixing things that aren't working right. There might be "bugs" in either your LEGO® machines or your Logo procedures (page 22).

force: A push or pull that causes an object to speed up or slow down. Gravity and friction are examples of forces (page 6).

friction: A force that causes an object to slow down. Rough surfaces cause more friction than smooth surfaces.

gear: A wheel with "teeth." You can use gears to change the position, strength, speed, and direction of a rotating motion (pages 6, 40-43).

gear train: A combination of gears used for changing the strength and speed of a rotating motion. When a small gear "meshes" with a large gear, the large gear rotates more slowly, but with greater strength (page 40).

gravity: A universal force that pulls all objects toward one another. The Earth's gravity pulls all objects down toward the center of the Earth.

infrared: A type of light beyond the red end of the rainbow. The LEGO® optosensor sends out and detects infrared light (page 45).

interface box: The box that connects LEGO® motors, lights, and sensors to the computer. The interface box has 12 ports where you can plug in motors, lights, and sensors (page 9).

list: A special type of Logo object that combines several pieces of information into a single unit. Lists are enclosed in square brackets, like `[1 2 3]`. By using a list, you can manipulate many pieces of information at once, rather than separately (page 17).

motor: A machine that provides a rotating motion. LEGO® motors convert electrical energy to mechanical energy (page 6).

optosensor: A sensor that detects changes in light. The LEGO optosensor usually acts like a "dark sensor." It reports `true` when it "sees" something dark (page 28).

port: A set of holes in the interface box where you can plug a LEGO® wire. Some ports (**outlets**) send messages to motors and lights. Other ports (**inlets**) receive messages from sensors (page 9).

primitive: A procedure that is already built into the Logo language (page 21).

procedure: A named set of Logo instructions. When you type the name of a procedure in the Command Center, the computer runs the procedure (page 20).

Procedure Center: The place where you define procedures. The Procedure Center is at the top of the computer screen (page 21).

recursive procedure: A procedure that includes itself as part of its definition. Some recursive procedures run through the same "loop" over and over (page 26).

sensor: A device that sends information about the world to the computer. LEGO® TC logo has touch sensors and optosensors (page 8).

software: Instructions that tell the computer what to do. Software is often stored on a **disk**.

subprocedure: A procedure that is used within the definition of another procedure (page 24).

theory: A set of beliefs or guesses about the way things work (page 5).

LEGO® TC logo

LEGO Dacta Educational Products
555 Taylor Road
Enfield, CT 06082

LEGO Australia Pty. Ltd.
P.O. Box 639
Lane Cove
N.S.W. 2066

ISBN 0-914831-69-0
198324

Printed in U.S.A.